

APPENDIX I

```
objectMolecule(String moleculeCallerName, String moleculeCalledName)
{

    /* Contributes to methods, methodTriggers, communication */
    objectNames[]           = Object names ( session/client names )
    objectContexts[]        = Equivalent to package ( molecule is compoante to
                             a function
                             collection of molecule, and is of relative Context)

    /* Socketing / Sessioning */
    objectContextKey{}      = controls scope of access to event, field, and method
                             groups, names, locations, version labeling, establish
                             status,

    {

        eventNeighbors{}    = List of comsuming molecules    (pass
                             Events/Data)
        eventNeighborsConsumeRoute{} = List of which Local/Remote data sets the neighbor
                             molecules can consume (Routing);
        eventNeighborsProduceRoute{} = List of where then consuming routing produces the
                             data sets local to the neighbor molecule;

        /* The objectMolecule.Deamon.thread should maintain a record of molecule channel
        IO*/
        fieldChannel{}       = Table of which channels are active
        fieldChannelExposed{} = Table defining channels inter/external exposablity
        fieldLocalStatic{}   = Table of static local memory routing
        fieldLocalStaticExposed{} = Table of defining static local memory routing
                             exposability
        fieldLocalDynamic{}  = Table of dynamic local channel routing
        fieldLocalDynamicExposed{} = Table of defining local channel exposability
    }

    /* localRemoteSwitcher() */

    objectMethodAccessLabeling[] = Determines Local or Remote access for methods
    objectMethod[]               = Methods callable name (local to Java or via API)
    objectMethodMoleculeData{}  = Data from File.moleculeName that forfills
    Consuming/Producing Routing.
```

/*methodTrigger() data , Local/Remote flags*/

objectWait[]	= methodTrigger() lifespan, or lifespan dependancy
objectMethodConsumingRouting[]	= Which Channel/Local data have methodTrigger() block on
objectMethodConsumingFofillment[]	= Where a methodTrigger() can look for data local and or remote and if it can switch between the two and if so who takes priority.
objectMethodProducingRouting[]	= Which Channel/Local space data is delivered to.

a methodTrigger().thread is started for each objectMethod[] element. All interfactable molecule should have enough data to create a methodTrigger for objectDisplay(). fieldChannel.LocalStatic.LocalDynamic{ } . a localRemoteSwitcher().thread is started as a child to methodTrigger().thread as appropriate, which is governed by objectMethodAccessLabeling[],objectMethodConsumingFofillment[]

}

objectDisplay()

{

struct objectDisplayEnvi {

displayEnviroment[]	= Reference to current interface environment, track
grounding, Screen tracking system	
displayState{ }	= Switch modes Accept Events, Accept Args, Accept Pointer, .. etc Criteria relevant object/ Superscope/subscope
displayAlphas[]	= zBuffersLayer's relative alpha (range 0.0 - 1.0) array allows
single object to multiple Depths	or volume Depth assignment.
displayZDepths[]	= zBufferLayer assignment, array allows single object to
multiple Depths	or volume Depth assignment.
displayZElements[]	= micro layering for a given zBufferLayer
displayComposites[]	= Object accumulation (add, subtract, dif, ..)
displayLocations[]	= Global location when parent, relative location when child

regPeriod[] = Array to (x,y image) or (x,y,z volume)

regDisplay{ } = Display scale as opposed to preObjectArray x,y,and/or z dimensions and registration

Array[] = array from an ObjectDisplayEngine()

EventArray[] = array from an EventDisplayEngine()

RegArray[] = array from an RegistrationDisplayEngine()

}

}

objectEnviroment()

{

Establish broadcasting pointer information channels and display through put channels.
}

-----Core Array Process Methods-----

Type: Data/String

MultiArray = array with N dimensions

SinglArray = arrayX1,arrayX2,...,arrayXN

MixedArray = arrayX1,arrayX2,array with 1-N dimensions,...,arrayXN

Algorithm:(API call, Java Class/Method Call)

Encode:

- consumer(type,MixedArray,format) produce(file)
- *raw to format (simple to many)
- *Multipl SinglArrays to aif,wav,txt
- *Multipl SinglArrays, or MixedArray to rgb,tif,jpg

Decode:

- consumer(file) produce(type,MixedArray)
- *format to raw (many to simple)
- *aif,wav,txt to Multipl SinglArrays
- *rgb,tif,jpg to Multipl SinglArrays, or MixedArray

Cmd Line:

- pass(String cmd,String argv[]) return(pid,status)

Splitter:

- consume(type,MixedArray[N],how) produce(type,MixedArray)

Mixer:

- consume(type,MixedArray[N],how) produce(type,MixedArray)

Stream:

- consume(type,SinglArray,how,rate) produce(channel VARstream)

PacketStream:

- consume(type,MixedArray,how,rate) produce(channel VAR[]stream)

Packet:

- consume(channel VARstream,how) produce(type,SinglArray)

MultiPacket:

- consume(channel VAR[]stream,how) produce(type,MultiArray)

Formula w/ Logic controlling output:

- consume(Elm & MixedArray,Sting equation) produce(Elm & MixedArray);
- *Composite Over,under,multiple,add,difference

Range:

- consume(type,MixedArray[N]) produce(type,min[X1->N],max[X1->N],avg[X1->N])

Sort:

- consume(type,SinglArray,key) produce(type,SinglArray)

Element:

-consume(type,MixedArray[N],index[]) produce(type,MixedArray)

Crop:

-consume(type,MixedArray[N],indexStart[X1->N],indexEnd[X1->N])
produce(type,arrayX1->N)

Scale:

-consume(type,MixedArray[N],type,Factor) produce(type,MixedArray[N])

Sample:

-consume(type,MixedArray,step) produce(type,MixedArray)

Flip:

-consume(type,SinglArray,xdirection,ydirection) produce(type,SinglArray)

Filter:

-consume(type,SinglArray,) produce(type,SinglArray)

Rotate:

-consume(type,SinglArray,angle,direction) produce(type,SinglArray)

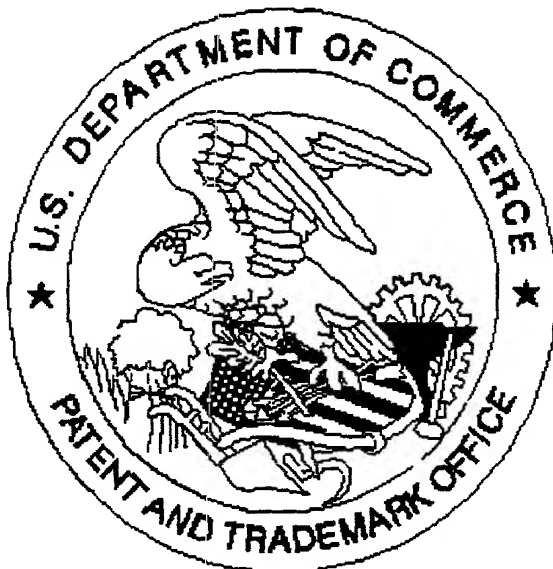
Interpolate:

-consume(type,MixedArray[N],interp_type) produce(type,MixedArray[N])

Tasks:

Data manipulation Process sequencing tool

United States Patent & Trademark Office
Office of Initial Patent Examination -- Scanning Division



Application deficiencies found during scanning:

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

there are 42 pages of specification, 9 pages of drawings
available for scanning.

☐ *Scanned copy is best available.*